Object-Based FEA Modeling in IMOS
(A progress report)

Greg Moore
JPL

FEMCI 2002,
May 23

JPL

Interferometry Center of Excellence

ICE

Outline

- IMOS (Integrated Modeling of Optical Systems) one year ago
- Development goals
- Some notes on object-based design and large-scale FEA
- Implementation in IMOS
- Future development

What is IMOS?

- Toolbox of Matlab script and executable files (*.m, *.mex) for finite element structural and thermal analysis, optical ray-tracing, statistical energy analysis, limited pre/postprocessing.

- Much of IMOS' flexibility is due to Matlab-hosted environment
  - matrix utilities, numerics, controls, visualisation
  - other tools running within Matlab environment (e.g. MACOS)

- (picture or avi of recent app.?)

IMOS status, one year ago

- At FEMCI 2001 we outlined some of IMOS' shortcomings, and motivation behind proposed major code overhaul:
  - lack of large problem scalability (~50k dof has been practial limit)
  - flexibility, ease-of-use in conflict
  - little support for higher-level analysis and design concepts
  - minimal data recovery, postprocessing

Motivation for recent, and future work

Previous shortcomings, especially:

- large problem scalability and performance
- support for NASTRAN models without data translation (native NASTRAN interface)
- Support for higher-level concepts (e.g. case, or state, control, substructures, multiple configurations (boundary conditions)
- Enhanced multidisciplinary analysis capabilities
  - integration with MACOS
- End-to-end design sensitivities and optimization

Some technical considerations in code redesign

- Efficient use of computing resources
  - compute space to span local memory, disk, and remote machines
  - indirect addressing (elimination of namespace collision)

```
for every substructure {
   for every boundary condition {
      compute reduced stiffness, mass matrix
   }
}
```

  - static, vs. dynamic, objects
- computational efficiency
  - maximum code reusability, minimal code overhead
  - NASTRAN model description compatibility, data structure and functional compatibility

JPL

Technical considerations, cont.:

- scripts-data-code:
  - user convenience != programmer convenience
  - how much functionality goes in Matlab, how much in executable code?
  - Where should data reside, and in what form?

Some Definitions:

- Object-based, vs. object-oriented code:

Object-oriented:

Object-based:

abstraction	Creation of a well-defined object interface
encapsulation	Keeping abstraction details hidden
heirarchy	Ability to reuse abstractions
polymorphism	Methods transparency for derived objects

Thinking about FEM objects:

- Is the finite element model:
  - The set of mathematical operations and approximations
  - The collection of grid points, elements, discretized loads etc.?
  - The resulting matrices?
- Answer:

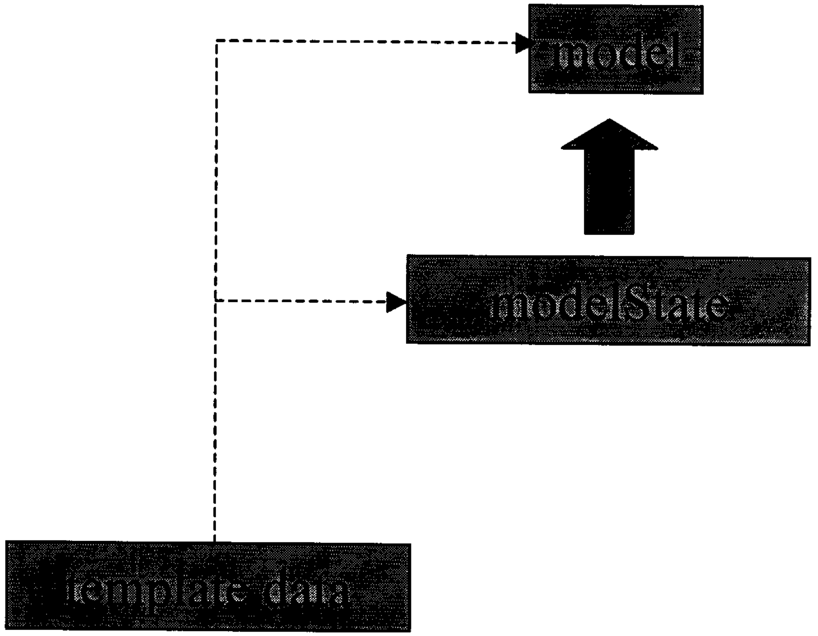
  YES

Thinking about FEM objects, cont.:

Object-based approach must also accommodate:

- Substructures/assemblies
- Multiple boundary conditions
- Modeling conventions
  - (autospc, dynamic reduction, inertia relief, etc.)
- Design states (model parameterization)

IMOS model containers:

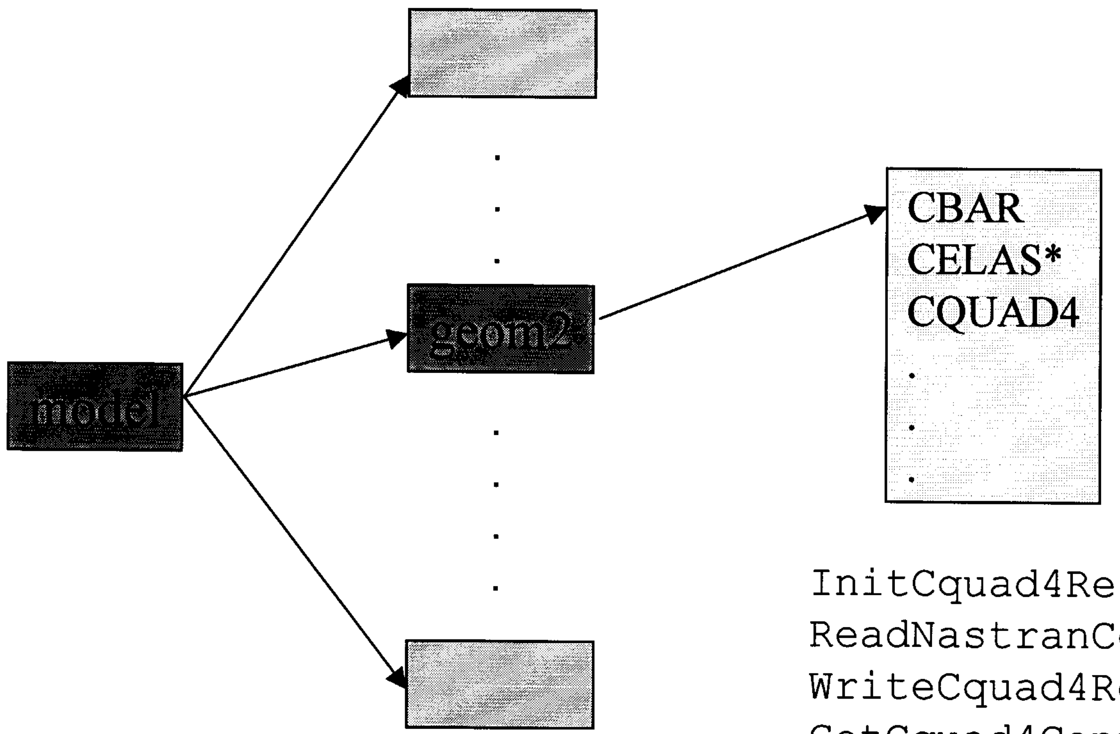


Basic, invariant data

Model configurations

User-modifiable in Matlab

Container classes:



```
InitCquad4Record();
ReadNastranCquad4Record();
WriteCquad4Record();
GetCquad4Connectivity();
```

■	Matlab-resident data
□	Compact (bytestream) data, location depends on template

Benefits:

- Container classes provide top-down, heirarchichal framework for complex model data
- Matlab-based template information provides user modifiability
- Namespace collision avoided
- Abstraction/encapsulation ensures dataset, individual data element integrity
- Open source provides unlimited customization

Selected new functionality:

FEA model reader:

```
[arrayOfModels, arrayOfModelStates] = . . .
                IMReadInputFile(infile,nullfile);
```

- Matlab executable (*.mex) for speed
- Native NASTRAN input (STEP extensible)
- Small, large, and free-field support
- Unlimited model sizes, continuations
- Case control (states), substructures, in progress

Interferometry Center of Excellence

ICE

Selected new functionality, cont.:

Data extraction:

```
[data_to_workspace] = . . .
              Imdb('verb object from dataset where clause');
```

Example:

```
[ni] = IMdb('select ni from geom2 where name=cquad4);
```

- Matlab executable
- Data either in memory or on disk (could be remote, too)
- Based on data structure api's
- Performance is excellent

Example:

```
                        < M A T L A B >
              Copyright 1984-1999 The MathWorks, Inc.
                    Version 5.3.1.29215a (R11.1)
                            Oct  6 1999
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, type tour or visit www.mathworks.com.

>> IMDataStruct;
>> infile = 'ngst_concept.dat';
>> nullfile = 'ngst_concept.null'
>> [ept,geom1,geom2,ifs,mpt] = IMReadInputFile(infile,nullfile);
   Input file summary:
   bulk data entry count:
    cbar             : 3065
    cord*            : 100   (includes all cordx*-type records)
    cquad4           : 76228
    crod             : 5296
    ctria3           : 32116
    grid             : 81342
    mat*            : 71    (  71 mat1's  )
    pbar            : 31
    prod            : 2
    pshell          : 97

created data sets
registered data sets
wrote data sets

>> [xyz] = IMdb('select xyz from geom1');

>> whos
  Name                         Size          Bytes  Class

  IMOSDataPath                 1x3            1972  struct array (global)
  IMOSDataStruct               1x12           4922  struct array (global)
  IMOSDefaultLocation          1x10           2054  struct array (global)
  ept                          1x1             534  struct array
  geom1                        1x1             538  struct array
  geom2                        1x1             538  struct array
  ifs                          1x1             534  struct array
  mpt                          1x1             534  struct array
  xyz                      81342x4         2602944  double array

Grand total is 325899 elements using 2614570 bytes
```

Current/Future work:

- Driven by design modeling considerations, e.g.:

$$\nabla P(x,t) \rightarrow \nabla u(x,t) \rightarrow \nabla u^{S}(x,t) \rightarrow \nabla PSF(x,t)$$

- MACOS interface
- NASTRAN element set migration
- FEA-based conductive and radiative heat transfer